

# Journal of Electronic Imaging

SPIEDigitalLibrary.org/jei

## **Development and simulation of soft morphological operators for a field programmable gate array**

Andrew J. Tickle  
Paul K. Harvey  
Jeremy S. Smith  
Q. Henry Wu



# Development and simulation of soft morphological operators for a field programmable gate array

Andrew J. Tickle

Coventry University

Department of Aerospace, Electrical, and Electronic Engineering

Priory Street Coventry CV1 5FB, United Kingdom

E-mail: [andrew.tickle@coventry.ac.uk](mailto:andrew.tickle@coventry.ac.uk)

Paul K. Harvey

Jeremy S. Smith

Q. Henry Wu

University of Liverpool

School of Electrical Engineering, Electronics, and Computer Science

Brownlow Hill Liverpool L69 3GJ, United Kingdom

**Abstract.** In image processing applications, soft mathematical morphology (MM) can be employed for both binary and grayscale systems and is derived from set theory. Soft MM techniques have improved behavior over standard morphological operations in noisy environments, as they can preserve small details within an image. This makes them suitable for use in image processing applications on portable field programmable gate arrays for tasks such as robotics and security. We explain how the systems were developed using Altera's DSP Builder in order to provide optimized code for the many different devices currently on the market. Also included is how the circuits can be inserted and combined with previously developed work in order to increase their functionality. The testing procedures involved loading different images into these systems and analyzing the outputs against MATLAB-generated validation images. A set of soft morphological operations are described, which can then be applied to various tasks and easily modified in size via altering the line buffer settings inside the system to accommodate a range of image attributes ranging from image sizes such as  $320 \times 240$  pixels for basic webcam imagery up to high quality  $4000 \times 4000$  pixel images for military applications. © 2013 SPIE and IS&T [DOI: [10.1117/1.JEI.22.2.023034](https://doi.org/10.1117/1.JEI.22.2.023034)]

## 1 Introduction

This paper continues the research described in Refs. 1 and 2, whilst still using the methodologies presented in them to implement the circuitry/algorithms and give them greater noise immunity to make them more robust. The system was designed in order to allow the  $k$ -value, as it is called here, to be easily changed without the internal setting having to be changed. This  $k$ -value is what gives the system its noise immunity. When  $k = 1$ , the system will operate in the same way as standard morphological operations (MO), but as this value is increased, the noise immunity is subsequently increased but at the cost of losing some of the image

data. One target application of this work can be soft morphological operation (SMO) filters, which are a combination of standard MO filters and a weighted rank order filter,<sup>3,4</sup> having the potential for very high performance image processing. Other possible applications include using mathematical morphology (MM) to remove noise to improve the quality and performance of automated imaging systems. These include tracking for example,<sup>5,6</sup> by effectively removing the data that is not required in the background or the noise in the foreground better than standard MOs. These processes could further assist in encryption and application specific systems developed for field programmable gate arrays (FPGA) using this methodology in conjunction with hand-written code.<sup>7–10</sup> As with previous systems, there will be two  $3 \times 3$  pixel structuring element (SE) cases used to perform this work on  $640 \times 480$  imagery. These circuits are able to compute the output image files for a given input file with a given  $k$ -value in one time step per pixel. In order to save time and resources during the advanced calculations of a single  $k$ -value, it would be wise to perform these operations with a single pass, rather than a cascaded effect of erosion and dilation or vice versa.<sup>11</sup> Without the single step operation, it would require two processing passes for one advanced operation (either opening or closing), which produces a delay in the filtering process and limits the speed at which the system can be run. This cascaded process requires large amounts of memory in order to store the intermediate result, which is why the cascade effect is not an efficient method since the programmable interconnects and memory on an FPGA is limited.

These SMOs allow the user to perform cascaded operations in the previous multifunctional system with different  $k$ -values, which was one of the reasons why the system presented in Ref. 2 was designed that way. Presented in this paper are the optimal system configurations and layouts, while briefly discussing the alternatives and the reasons why they were not suitable for the applications at hand. Many computer vision applications can benefit from the integration of image capture and image processing functions

Paper 10053 received Aug. 22, 2010; revised manuscript received Aug. 6, 2012; accepted for publication Apr. 29, 2013; published online Jun. 25, 2013.

0091-3286/2013/\$25.00 © 2013 SPIE and IS&T

onto an FPGA by describing the gate level implementation in a layered structure in order to reduce delays and to aid in debugging operations.<sup>12,13</sup> The two main layouts were a multi- $k$  SMO system and a separated- $k$  SMO system. With the multi- $k$  system, all the data goes into one block and all copies and pixel values are ranked and sorted before being correctly output to the right ports. The separated- $k$  SMO system, on the other hand, has a block which performs each  $k$ -value operation separately in a parallel processing architecture<sup>14</sup> rather than a single block system. An overall decision block then decides which signals are output based on the  $k$ -value input via the user whilst the remaining signals are terminated. A size comparison also takes place where the circuits from Ref. 1 are compared to the ones developed in this paper to determine if they take up more, less, or roughly the same amount of gates as the previous systems. This comparison will also indicate whether the systems can still be synthesized onto an FPGA or not, it also provides an idea of the robustness of the SMOs against the MOs as when  $k = 1$  the output from both these systems should be the same. Furthermore, the size of the developed circuitry should try to be as close as possible to the ones from Ref. 1 so that the  $k$ -value can be varied to higher values to provide greater noise immunity without the associated proportional increase in the size and complexity of the circuitry. A pixel-by-pixel analysis is also performed on the images and the results from noisy images are also discussed. The latter were passed through the system to investigate how noise affects the results and the comparison process. The work here is to present the most basic SMOs at the two most commonly used levels/dimensions so that work can then continue

into the more advanced SMOs and their associated systems and applications.

Section 2 presents the mathematical morphology (MM) theory which is structured into circuitry and described in Sec. 3. Computational complexity, effectiveness of the circuit operations, and size comparisons are analyzed in Sec. 4, whilst conclusions are drawn in Sec. 5.

## 2 Soft Morphological Operations

Here, the SE is divided into two parts: the core which has a weight or number of copies  $k$  (repetition parameter) which can be equal to or greater than “1” and the soft boundary which acts in the same way as the pixels in a standard MO structuring element. This is defined mathematically as the core and the soft boundary of the SE  $B$  by  $B_1$  and  $B_2$ , respectively,  $B = B_1 \cup B_2$ . It has been proven that soft morphological operations are advantageous regarding additive noise elimination and sensitivity to small variations in object shape when compared to standard morphological transforms.<sup>3,11</sup> The shape for SE4 (a 4-connectivity SE which is defined in Ref. 1) is shown where the bold numbers represent the core of the SE. Their corresponding pixel values are also shown here:

$$SE4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \text{Pixel 1} & \text{Pixel 2} & \text{Pixel 2} \\ \text{Pixel 4} & \text{Pixel 5} & \text{Pixel 6} \\ \text{Pixel 7} & \text{Pixel 8} & \text{Pixel 9} \end{bmatrix}. \quad (1)$$

### 2.1 Binary Erosion

Binary soft erosion is defined as follows,<sup>15</sup> where  $\text{Card}[X]$  represents the cardinality or number of elements of the set  $X$ :

$$(A \ominus B_1, B_2, k)(x) = \begin{cases} 1 & \text{if } (k \cdot \text{Card}[A \cap (B_1)_x] + \text{Card}[A \cap (B_2)_x] \geq k \cdot \text{Card}[B_1] + \text{Card}[B_2] - k + 1) \\ 0 & \text{Otherwise} \end{cases}. \quad (2)$$

Equation 2 states that in order to softly erode a binary image set  $A$  by a binary SE  $B$ , the SE is moved over the image set applying a standard MO. At each point  $x$ , the pixel intersection of  $A$  is a “1” with the translated core  $B_1$  must be found and multiplied  $k$  times, then added to the cardinality of the intersection of  $A$  with the translated soft boundary  $B_2$ . The same operation is performed for the right-hand side of the equation. If this result is greater than or equal to  $k \cdot \text{Card}[B_1] + \text{Card}[B_2] - k + 1$  then the output pixel for that  $3 \times 3$  pixel segment is a “1.” There is some more information and examples of this process in Ref. 16.

### 2.2 Binary Dilation

The process for binary soft dilation is very similar to erosion apart from the equation for the operation, which now becomes:

$$(A \oplus B_1, B_2, k)(x) = \begin{cases} 1 & \text{if } (k \cdot \text{Card}[A \cap (B_1)_x] + \text{Card}[A \cap (B_2)_x] \geq k) \\ 0 & \text{Otherwise} \end{cases}. \quad (3)$$

### 2.3 Grayscale Erosion

Here,  $k \diamond x$  denotes the  $k$ -times repetition of  $x$ . The core and the soft boundary of the SE  $g(z)$  are denoted by  $\alpha(z_\alpha)$  and  $\beta(z_\beta)$ , respectively:

$$z_\alpha \in D[\alpha], \quad z_\beta \in D[\beta] = D[g]/D[\alpha], \quad (4)$$

where  $D[g]/D[\alpha]$  denotes the set difference. The soft erosion of  $f$  by a SE  $g$  with a core  $\alpha$ , and a soft boundary  $\beta$  is defined in Ref. 16:

$$(f \ominus [\beta, \alpha, k])(x) = \min^k(k \diamond \{f[z_1 - \alpha(z_1 - x)]\} \cup f[z_2 - \beta(z_2 - x)]) \quad (5)$$

$$z_1 - x \in D[\alpha] \quad (6)$$

$$z_2 - x \in D[\beta] = D[g]/D[\alpha], \quad (7)$$

where  $\min^k$  is the  $k$ 'th smallest of the multiset in the parentheses, which is a multiset collection of objects where

repetition is allowed. In order to grayscale soft erode an image set  $f$  by a SE  $g$ , the SE is moved over the image set as described previously, finding all the differences of the values of  $f$  with their corresponding values of the translated core  $\alpha$ , after which all the differences of the values of  $f$  with the corresponding values of the translated soft boundary  $\beta$  are produced. The resulting values are ranked in ascending order including the  $k$  copies of the core  $\alpha$  and the  $k$ 'th smallest is selected as the result.

## 2.4 Grayscale Dilation

The soft dilation of  $f$  by an SE  $g$  also with core  $\alpha$  and soft boundary  $\beta$  is defined in Ref. 16:

$$(f \oplus [\beta, \alpha, k])(x) = \max^k \{k \diamond [f(z_1) - a(x - z_1)] \cup f(z_2) - \beta(x - z_2)\} \quad (8)$$

$$x - z_1 \in D[\alpha] \quad (9)$$

$$x - z_2 \in D[\beta] = D[g]/D[\alpha], \quad (10)$$

where  $\max^k$  is now the  $k$ 'th largest of the multiset in the parentheses. The soft dilation of  $f$  by  $g$  is translated spatially in the reflection of  $g$ ,  $g(-z)$ , so that it is located at the starting point, after which the same steps are followed as in the soft erosion case, apart from the answer now producing the  $k$ 'th highest instead of lowest.

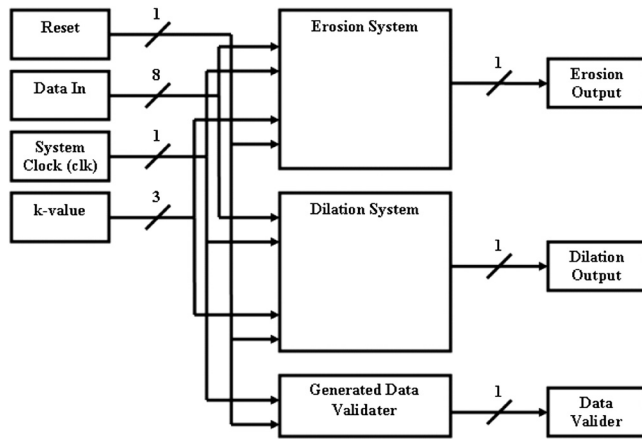


Fig. 1 Register transfer level (RTL) model for the binary system.

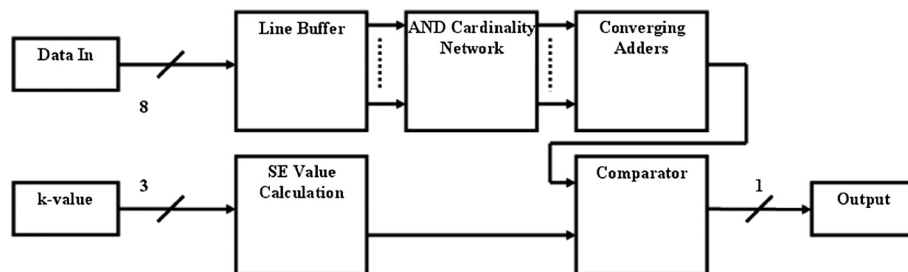


Fig. 2 Gate level model for the morphological calculator for the binary system.

## 2.5 Soft Opening and Closing

Soft opening and soft closing for both binary and grayscale systems are defined in Ref. 5:

$$f \circ [\beta, \alpha, k] = (f \ominus [\beta, \alpha, k]) \oplus [\beta, \alpha, k] \quad (11)$$

$$f \cdot [\beta, \alpha, k] = (f \oplus [\beta, \alpha, k]) \ominus [\beta, \alpha, k]. \quad (12)$$

As with the standard morphological operations, the soft opening is a dual to soft closing and vice versa, but now unlike standard opening and closing they are neither extensive nor anti-extensive. In general, both soft opening and soft closing are not idempotent, and it has been proven that soft opening and closing are idempotent only under specific conditions. For example, in the case that  $g = g^\delta$ , the core includes only one pixel which is located at the origin.<sup>16</sup>

## 3 Development of Systems

### 3.1 Binary Soft Morphology Circuitry

The system was designed so that the  $k$ -value is a constant throughout the system and therefore does not change. Figures 1 and 2 show two levels of abstraction for the binary system. The inputs and outputs of Fig. 1, on the left and right, respectively, show the data input, clock, global reset, and  $k$ -value input. These would be the inputs and outputs of the system at the behavioral level. Also shown is the representation of the signals being directed to the erosion calculator, dilation calculator, and data validator blocks, while Fig. 3 shows the basic layout of the dilation and erosion calculators. The SE value calculation block is the only section of this circuit which changes between the two systems, as this represents the RHS (right-hand side) of Eqs. (3) and (4).

The two calculator circuits start by having the  $3 \times 3$  image buffers which delay the image set in order to produce the  $3 \times 3$  image segment that the SE would move over if this were calculated by hand.<sup>1</sup> The soft pixel part of the SE is sent into a series of AND gates with a “1” going into the other input port so that if the input from the image segment is also a “1,” then the output from the AND gate will be a “1” and help calculate the cardinality of the soft part of the SE. Thus, this is the AND cardinality block. These outputs are then added together using three layers of adders (converging adders block) before a final adder adds the  $k$ -value to this in order to take into account the cardinality of the core of the SE. This latter value is obtained by sending the signal for pixel 5 from the image buffer into a product block, which is linked to the  $k$ -value input. So if pixel 5 is a “1” then the product will output the  $k$ -value to be added but if it is “0” then a



“0” will be output instead. Finally, this total value is then sent into a comparator and compared with the SE value calculation block output which, if greater than “1,” defines the result as “1” otherwise a “0” is the result.

The SE value calculation block of the system either adds the number of soft output pixels which comes from a constant block in order for them to be more easily changed (similarly from SE8 (an 8-connectivity SE, defined in a similar manner to SE4) to SE4) by changing the constant from an “8” to a “4”) to the  $k$ -value which represents the core. This is then added to a “1” value and then the  $k$ -value is subtracted for the erosion case whereas for dilation, the  $k$ -value is sent directly to the comparator at the end.

### 3.2 Grayscale Soft Morphology Circuitry

The behavioral level for the grayscale system is virtually identical to the binary system apart from the output data

line widths having been expanded to 8 bits in order to account for the grayscale values that will be dealt with. Figures 3–5 show the register transfer level (RTL) and gate levels of abstraction for the grayscale system for the separated- $k$  SMO system, where Fig. 3 shows the series of blocks moving from left to right, which calculates the  $k = 1$  to  $k = 7$  values, respectively, and sends them to the decision-maker block. In this block the correct  $k$ -value is finally output based on the  $k$ -value input into the system. The first block is an image buffer<sup>1</sup> that is modified slightly so that a “1” is either added or subtracted to all pixels in the image segment. In addition to this, a check was also added in order to prevent overflow so that a “1” would not be added or taken away if the value were “255” or “0,” respectively. This was accomplished by using a standard DSP Builder comparator to see if the value matches one of these values and if it does match, a “1” will be produced. It will then be inverted using a NOT gate and fed into an AND gate with a constant of

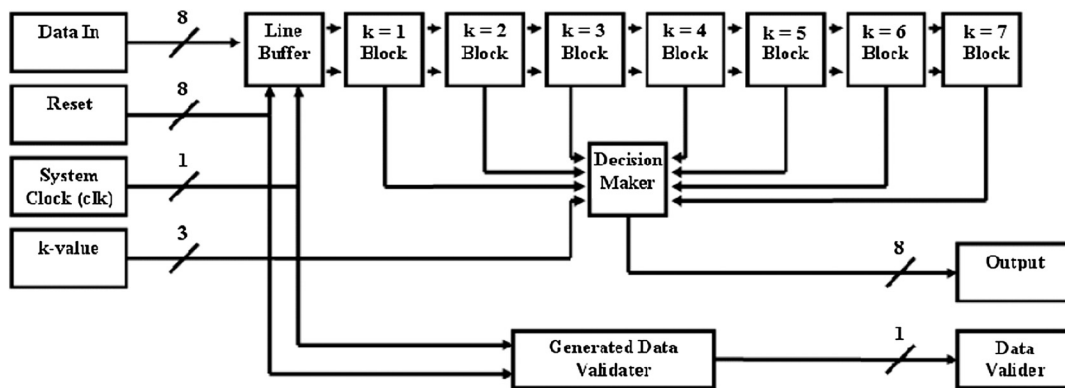


Fig. 3 RTL model for the grayscale system.

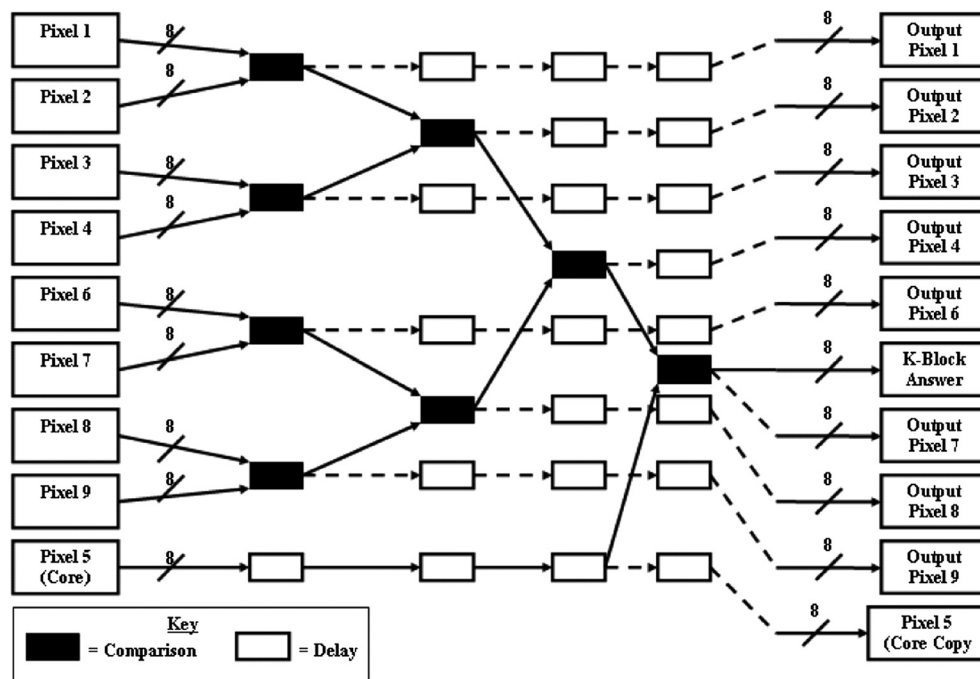


Fig. 4 Gate level model for the morphological calculator for the grayscale system.

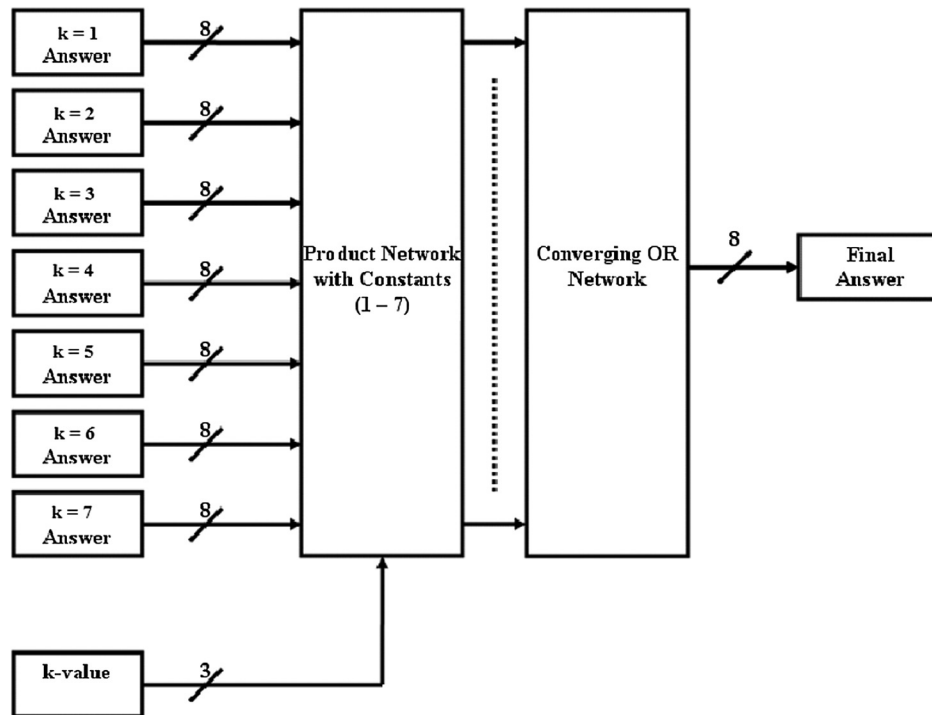


Fig. 5 Gate level model for the output decision block of the grayscale system.

“1” going in, so when they do not match they would both be “1”s and so the output from the AND would be a “1” which would then be added or subtracted, respectively. If they match however, the inputs to the AND would be a “1” and “0,” respectively, so that a “0” is produced and no changes occur.

Figure 4 shows the basic layout of the dilation and erosion calculators in which the main elements of the system are based around the bus comparator, previously developed in Ref. 1 but now working in pairs where one would find the highest, while the other finds the lowest. The reason for this is that the system finds the highest or lowest value through a process of comparisons, then outputs the highest or lowest (depending on the operation) as the answer. However, the values which were not the highest nor the lowest need to be retained which is why the opposite operation is used in order to do this so that these values can be passed onto the next block along with a copy of the core. This effectively removes the highest or lowest value pixel and so the same block can be repeatedly used for the different  $k$ -values instead of having to find the second highest, third highest, etc. and have a separate block for each. If the highest is removed, then when the process is repeated, the second highest will obviously now become the highest. The other single comparator blocks (the delay blocks shown in Fig. 4) are there to ensure that all signals remain in synchronization with each other, such that propagation delay does not cause a major problem. These values are then sent to the next block relative to the last block, which is just a simple conversion with no need to output the nonhighest or lowest values.

Figure 5 shows the decision-maker that decides which output from each of the possible seven  $k$ -value answers is in fact the true answer. There are the seven inputs from

the various blocks in the system which are each sent into a product block and the other input is from a comparator. These have constants ranging from 1 to 7 for the respective  $k$ -value signals, and are compared to the actual  $k$ -value so the matching signal will produce a “1” from the comparator. When sent into the product, the output value is fed into an OR gate network which takes the signal to the output port. When the output from the comparator is a “0,” no value is sent out from the product block into the OR network, so it would not corrupt the true output value.

Overall, the separated- $k$  SMO system is better than the multi- $k$  SMO system because the separated blocks are smaller in size, less complicated, and less likely to have propagation signal errors in the output. The multi- $k$  system had a series of nine blocks to find the first, second, third, etc. highest pixels and the number of copies of that pixel value. These were found by a process of comparisons to find the highest or lowest value, and then this result is compared to all pixel values to see how many other pixels also share that value, then the sum is computed and the data is output to the ranker. Eventually, all the pixel values would be “0” and the system will stop. A location signal was also sent out so that the system would know where and how to reconstruct the image segment. The ranker is also a complex piece of circuitry with over 120 possible combinations, so as a result the output was prone to propagation delay and dithering. The incoming signals had to have  $k$  copies of themselves made and the locations added up to prevent them from exceeding 15. If the value did exceed 15, the system halted due to a computational error (this is because a  $3 \times 3$  pixel image segment with a core  $k$ -value of 7 can only give an end result of  $15 = 8 + 7$ ). The signals are then sent to their correct ranked location along with their associated number of copies to be sent to the correct output

via several OR gate networks ranging from 5 to 3 layers of OR gates in each with 15 to 7 possible inputs. This explains why the system is not suitable, not only due to the problems mentioned previously, but it was also too large. As stated earlier, there are limited resources on an FPGA, which is why the separated- $k$  system is preferred.

### 3.3 Development of Advanced Systems

For each section, a series of calculations were performed and these were analyzed in order to find trends and patterns. The purpose here is that by analyzing many common configurations in the  $3 \times 3$  image segment, trends for these configurations could give rise to a set of rules that would allow direct implementation of the advanced morphological functions rather than simply cascading the basic functions together as discussed earlier. A similar process was undertaken and the same patterns were found for both the SE4 and SE8 cases. A similar process can be carried out for other shaped and valued SEs, using this work as a template if another SE were required for a particular operation. Each case will now be considered along with the supporting mathematical proof.

### 3.4 Binary Soft Opening Circuitry

It was discovered from the calculations that because the erosion step takes place first, the image set is mainly reduced to zero apart from when the  $k$ -value is high, such as when  $k = 4$  and above. The latter is true for many configurations of the SE tested. The dilation step in the opening process is therefore a carbon copy of this intermediate set if it has been completely reduced to zero as there is no chance that the values can increase and since the center pixel is equal to the  $k$ -value. This is so due to the fact that it has  $k$  copies and thus passes to the output image set while the remaining pixels (unless they are the center one) do not have enough active pixels

surrounding them to pass above the “ $k$ ” threshold, so the intermediate becomes the output. Several example situations are shown here:

$$\bullet \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \Rightarrow \begin{cases} 1 & \text{if } k = 1 \rightarrow 7 \\ 0 & \text{Never} \end{cases}$$

$$\bullet \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} \Rightarrow \begin{cases} 1 & \text{if } k = 4 \rightarrow 7 \\ 0 & \text{if } 1 \rightarrow 3 \end{cases}$$

$$\bullet \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} 1 & \text{if } k = 5 \rightarrow 7 \\ 0 & \text{if } k = 1 \rightarrow 4 \end{cases}$$

This means that the circuit to perform binary soft opening is the same as the binary soft erosion circuit shown in the previous section depending upon which  $k$ -value is in use.

### 3.5 Binary Soft Closing Circuitry

In a similar manner to the calculations shown in the previous section, if the intermediate step in the closing process has all its pixels at “1,” then the output will be “1,” otherwise it will be “0.” Therefore, a method was required to see if all the intermediate pixels will become “1”s. This involved looking at the pixels surrounding the center pixel to determine if the center pixel in each case would become a “1,” then adding the values of all the surrounding pixels. For the two common SE cases, (SE8 = 9 and SE4 = 5) the center pixel is a “1” if it passes this SE value. The equation for soft binary dilation is shown in Eq. (4) and can thus be modified to show mathematically what is happening here:

$$1 \quad \text{if } \sum_{\text{pixel}=1}^{\text{pixel}=9} \begin{cases} 1 & \text{if } (k \cdot \text{Card}[A \cap (B_1)_x] + \text{Card}[A \cap (B_2)_x]) \geq k \\ 0 & \text{Otherwise} \\ 0 & \text{Otherwise} \end{cases} \geq 9. \quad (13)$$

Here, the investigation looks specifically at the SE8 system and some specific test cases that will prove that the aforementioned system specifications work correctly:

$$\bullet \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \Rightarrow \begin{cases} 1 & \text{if } k = 1 \\ 0 & \text{if } k = 2 \rightarrow 7 \end{cases}$$

$$\bullet \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} 1 & \text{if } k = 1 \rightarrow 3 \\ 0 & \text{if } k = 4 \rightarrow 7 \end{cases}$$

$$\bullet \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} 1 & \text{if } k \neq 1 \rightarrow 7 \\ 0 & \text{if } k = 1 \rightarrow 7 \end{cases}$$

The proof for each one of these is several pages long and has not been included here. At the time of designing the system, each pixel had to consider those surrounding it, so for each pixel situation the following combinations had to be considered:

- Pixel 1 uses pixels 1, 2, 4, and 5.
- Pixel 2 uses pixels 1, 2, 3, 4, 5, and 6.
- Pixel 3 uses pixels 2, 3, 5, and 6.
- Pixel 4 uses pixels 1, 2, 4, 5, 7, and 8.
- Pixel 5 uses *all* pixels.
- Pixel 6 uses pixels 2, 3, 5, 6, 8, and 9.
- Pixel 7 uses pixels 4, 5, 7, and 8.
- Pixel 8 uses pixels 4, 5, 6, 7, 8, and 9.
- Pixel 9 uses pixels 5, 6, 8, and 9.

Another significant issue was that if a group of 4 zeroes arise in the system in the following configuration  $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ , then this indicates that under no circumstances can the center pixel become a “1” in the overall answer. This would apply for the following pixel configurations which had to be looked for, and this provided an additional safeguard against incorrect answers.

$$\begin{bmatrix} \text{Pixel1} & \text{Pixel2} \\ \text{Pixel4} & \text{Pixel5} \end{bmatrix} \begin{bmatrix} \text{Pixel2} & \text{Pixel3} \\ \text{Pixel5} & \text{Pixel6} \end{bmatrix} \begin{bmatrix} \text{Pixel4} & \text{Pixel5} \\ \text{Pixel7} & \text{Pixel8} \end{bmatrix} \begin{bmatrix} \text{Pixel5} & \text{Pixel6} \\ \text{Pixel8} & \text{Pixel9} \end{bmatrix}. \quad (14)$$

Figure 6 shows the gate level designs for the binary soft closing system. The RTL level is again similar to Fig. 1, apart from the two operation blocks now opening and closing and the data validator being the same. The gate level system is based around the basic SMO designs where the data input feeds the line buffers in order to create the  $3 \times 3$  image segment of the image set that will be processed. A series of AND gates with a constant value of “1” going into one input was used to detect the cardinality of the pixel configuration. However, instead of just using pixel 5 (the center pixel) and those surrounding it, all the pixel situations mentioned will behave in the same way. All the pixels are added together and then subsequently added to the value of the pixel that is currently being taken as the center. This is because every pixel in turn acts as the center pixel for one situation. This is multiplied by the  $k$ -value before being added to the cardinality value from the other pixels, and before being compared to the  $k$ -value using a standard comparator. Either a “1” or “0” answer is produced for all nine pixels in the image segment; these are then ANDed together and passed through a second comparator whose threshold is set to a constant “9” and if the sum of all pixels is greater than this, then the output is a “1” and this is the final system answer to be output.

The circuitry converges from all the separate pixel groups with pixel 1 at the top and moving downward to pixel 9. At the bottom of the gate level design is the extra configuration check which is composed of four large four-input AND gates. These gates stand to additionally safeguard against the incorrect answer situations. Each input to these 4-input AND gates

is inverted in order to find out when the pixel considered is a “0.” If this condition is met then the output will be a “1” and so this is again inverted back to a “0.” This is then fed into a four-input OR gate, basic logic gates like two-input OR gates with both inputs tied together act as delays (the delay module block in Fig. 6) to keep the signal in synchronization with the data passing through the other parts of the system. A product block is placed between the output from the other part of the circuit and the output port so that if the pixels are not all zeros; a “1” would be passed from the output of one of the AND gates through the OR gate and delays into the product, thus allowing the system to output the data. If they are all “0”s then a “0” is passed to the product and no data is allowed to leave the block. This condition is represented by the output flow controller block in Fig. 6.

### 3.6 Grayscale Soft Opening Circuitry

Showing all the steps with all the calculations for even a small  $k$ -value of 7 would again take several pages and so the highlights of just one example case,

$$\left( f \circ \text{SE} = \begin{bmatrix} 12 & 7 & 24 \\ 216 & 180 & 15 \\ 27 & 160 & 18 \end{bmatrix} \circ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right),$$

are shown; the intermediate steps SE have also been omitted again for the same reason. The first matrix represents the intermediate value while the second matrix represents the final image set:

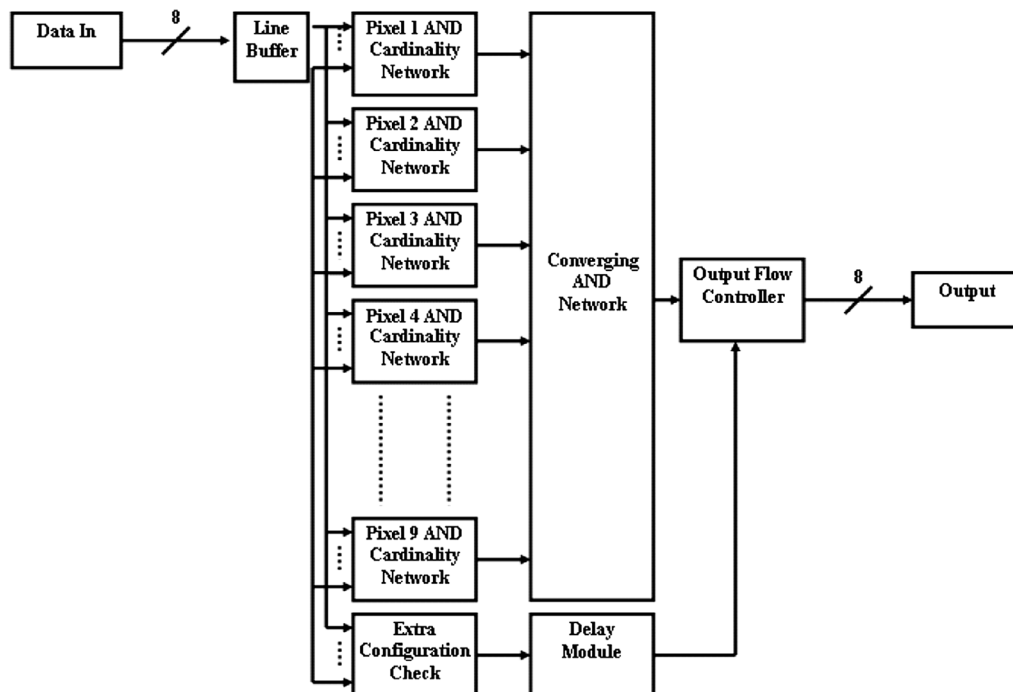


Fig. 6 Gate level view of closing circuitry.



- $k = 1 \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 7 & 7 & 7 \\ 7 & 7 & 7 \\ 7 & 7 & 7 \end{bmatrix}$
- $k = 3 \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 14 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & 15 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
- $k = 5 \Rightarrow \begin{bmatrix} 0 & 6 & 0 \\ 11 & 23 & 14 \\ 0 & 17 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 7 & 1 \\ 12 & 24 & 15 \\ 1 & 18 & 1 \end{bmatrix}$
- $k = 7 \Rightarrow \begin{bmatrix} 11 & 6 & 14 \\ 159 & 159 & 14 \\ 26 & 159 & 26 \end{bmatrix} \Rightarrow \begin{bmatrix} 12 & 7 & 15 \\ 160 & 160 & 15 \\ 27 & 160 & 27 \end{bmatrix}$

From this work, a clear pattern began to form: after calculating the initial stage, this rule can then be applied to calculate the final values which can be easily modified for the SE4 and therefore has not been shown:

- For  $k = 1$ : all pixels become the intermediate pixel and add one.
- For  $k = 2 \rightarrow 7$ : add one to all pixels in the intermediate result.

The process is very similar to the binary operations thus showing how similar techniques can be applied to higher dimensional imagery. Again, this can be represented mathematically by:

$$f \circ [\beta, \alpha, k] = (f \ominus [\beta, \alpha, k]) \oplus [\beta, \alpha, k] \\ = \min^k \{k \diamond [f(z_1) - \alpha_x(z_1)] \cup f[z_2 - \beta_x(z_2)]\} + 1. \quad (15)$$

This means that the circuitry to perform grayscale soft opening is the same as the grayscale soft erosion circuit, but with a “1” added to the result before the data is output.

### 3.7 Grayscale Soft Closing Circuitry

Here is a single case for grayscale closing:

$$\left( f \cdot SE = \begin{bmatrix} 211 & 27 & 31 \\ 199 & 31 & 34 \\ 201 & 32 & 30 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right),$$

but as can be seen from the data there is no direct pattern. Therefore, the intermediate value of each pixel had to be calculated from the surrounding pixels, then the basic SMO method is used to calculate the final value. This is almost a cascaded operation, but as it uses the values directly without any intermediate memory storage used, it is “technically” classed as a direct implementation method.

- $k = 1 \Rightarrow \begin{bmatrix} 212 & 212 & 35 \\ 212 & 212 & 35 \\ 202 & 202 & 25 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 34 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- $k = 3 \Rightarrow \begin{bmatrix} 212 & 35 & 32 \\ 200 & 200 & 35 \\ 202 & 33 & 32 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 32 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

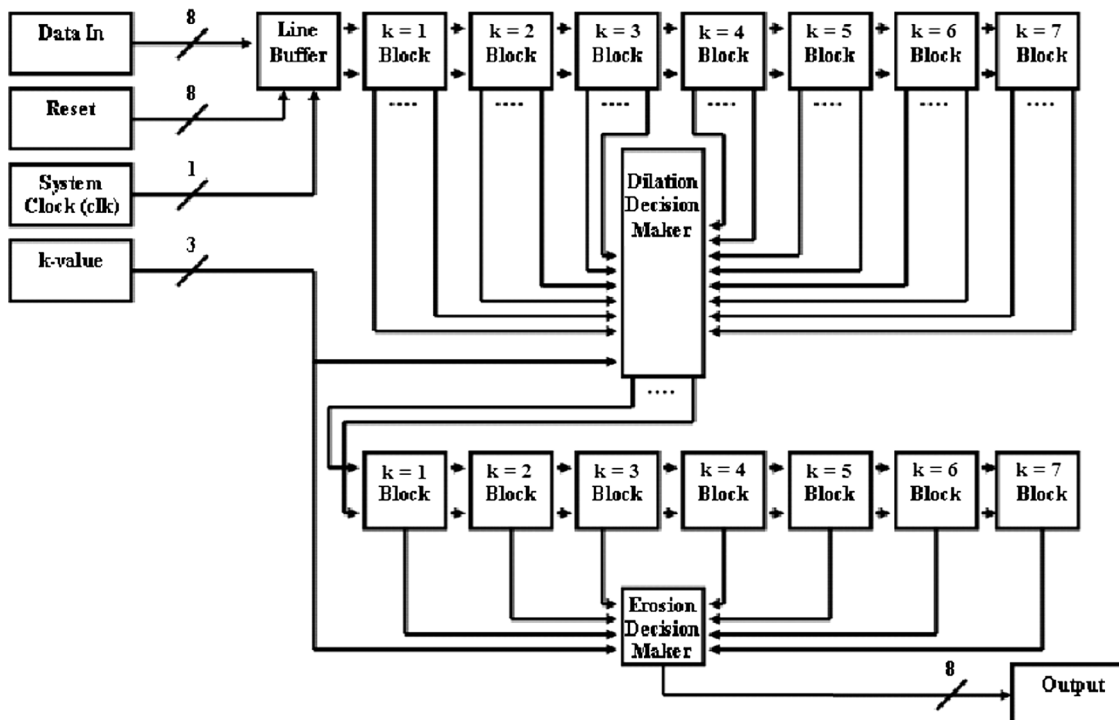


Fig. 7 RTL level for advanced soft grayscale operations.

$$\begin{aligned}
 \bullet \quad k = 5 &\Rightarrow \begin{bmatrix} 212 & 32 & 32 \\ 200 & 33 & 35 \\ 202 & 33 & 31 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 31 & 0 \\ 32 & 32 & 31 \\ 0 & 32 & 0 \end{bmatrix} \\
 \bullet \quad k = 7 &\Rightarrow \begin{bmatrix} 218 & 28 & 32 \\ 200 & 32 & 35 \\ 202 & 33 & 31 \end{bmatrix} \Rightarrow \begin{bmatrix} 31 & 27 & 31 \\ 199 & 31 & 31 \\ 32 & 32 & 30 \end{bmatrix}.
 \end{aligned}$$

Figure 7 shows the RTL level design for grayscale closing. The gate level blocks are very similar to the system used previously as in Fig. 4, where the bus comparators compare the signals for both highest and lowest values. In this case, the highest finds the answer for that particular  $k$ -value, while the lowest preserves the nonhighest values so that they can be used in the next block to find the next highest value. However, here a similar process is done as with the soft binary closing case where each pixel in turn acts as the center pixel, so each block outputs the nine highest pixel values for each of the nine pixel cases, while passing the unused data along with copies of the core(s) to the next block for the process to be repeated six times, in order to produce the seven  $k$ -value possibilities. These nine pixel values for each of the seven blocks are sent to the dilation decision maker block. This block works in a similar manner to the output decision block described previously.

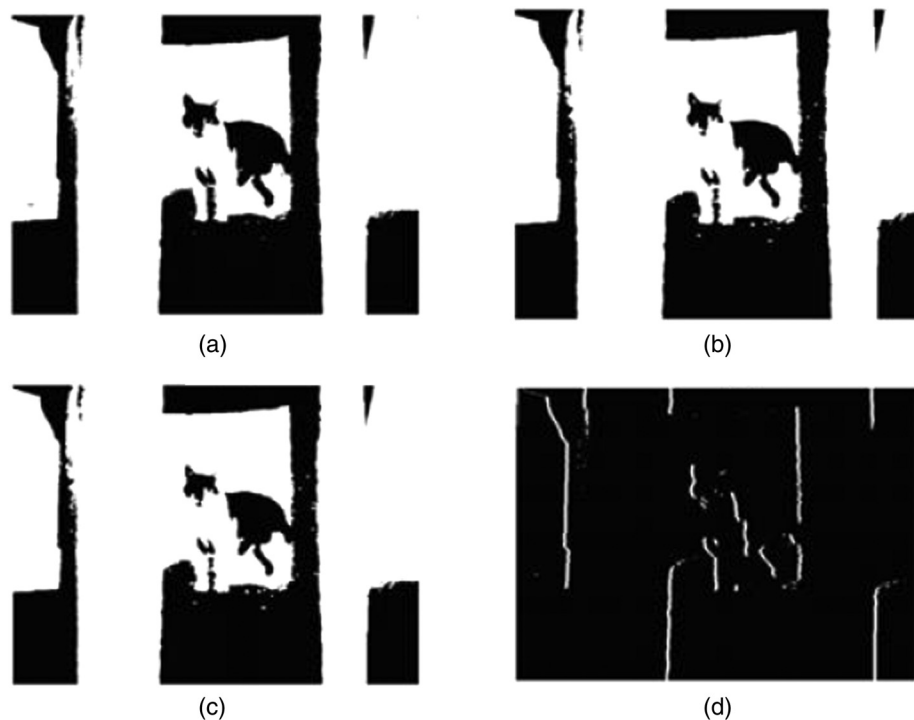
In this final output block, all inputs are connected to product blocks which are also connected to a basic comparator block where the pixels of each block have a constant value of “1” fed into one of the inputs of the comparator only when the other input has the  $k$ -value going into the system. This comparator output is connected to the product block and is repeated for all possible combinations of pixels. For example, if  $k = 1$  then the pixels from 1 to 9 for that case

will be allowed to continue to progress through the system, whereas all others will be turned into “0”s. Every like pixel is connected into the same OR gate network so that all pixels numbered 1 are connected together; all pixels numbered 2 are connected together, etc. Since only one will be active at any one time, the value will remain unaffected reaching the output of the network which is connected to the output port for that particular pixel. This ensures that the correct values are sent out to the next stage, which is the soft grayscale erosion block that performs the second stage in the closing process and is then output as the answer.

#### 4 Simulation Results, Size Comparisons, and Resource Allocations

Figure 8(a)–8(d) shows the results for the binary tests. All these tests were conducted with a  $640 \times 480$  color image that was converted to binary when sent into the DSP Builder simulation. The outputs from these systems were as in a previous work<sup>1</sup> and ran through a MATLAB m-file code to compare the number of pixels and locations to a check image for validity. This check image was produced using the MATLAB image processing toolbox as a starting point and then incorporating the equations shown earlier that govern the morphology to calculate the check images in order to get the higher  $k$ -values. These were calculated using the same sweeping pattern over the image that the DSP Builder simulation would use and was originally tested on smaller images of which could be calculated by hand to provide additional validity. Some of these hand calculations are shown in a cut-down capacity due to their length in Sec. 3 when the development of systems was discussed and the entire calculations can be found in Ref. 17.

The results show that the DSP Builder systems and MATLAB checks that the  $k = 1$  systems are identical [Fig. 8(b)



**Fig. 8** Simulation results for binary operations, (a) Binary counterpart of original image, (b) Morphological dilation  $k = 1$  result, (c) MATLAB dilation  $k = 1$  check image, (d) Morphological erosion/dilation  $k = 7$  result.

and 8(c) locations are identical] with all changes occurring in the same places. For the  $k = 7$  case, the two situations are very similar. This occurs in both binary and grayscale situations and is best described by looking at it from a ranked point of view. Dilation will be the highest. As the  $k$ -value increases it moves down the “ladder” of ranked values to lower values, whereas the erosion always stays at the same level on the “ladder” because a new copy of the core is always added with each  $k$ -value and so the lowest, second lowest, etc., always take place on the ninth highest “ladder” level. This is visualized below. The higher the  $k$ -value, the nearer the erosion and dilation output images will move towards each other. This will be more noticeable in binary images because there are only two possible data values, while with the grayscale system you can also tell a distinct difference between the  $k = 1$  and  $k = 7$  values, and how the similarity of the images become clearer.

- First ladder level = dilation where  $k = 1$
- Second ladder level = dilation where  $k = 2$
- Third ladder level = dilation where  $k = 3$
- Fourth ladder level = dilation where  $k = 4$
- Fifth ladder level = dilation where  $k = 5$
- Sixth ladder level = dilation where  $k = 6$
- Seventh ladder level = dilation where  $k = 7$
- Eighth ladder level = unused
- Ninth ladder level = erosion where  $k = 1 \rightarrow 7$
- Tenth ladder level = unused.

However, during testing there was a problem encountered with the binary erosion and so a bias value of “2” had to be added onto the existing  $k$ -value. This was because originally the output image set was saturated with zeros and upon examination of the output, it was discovered that the values in the output data stream ranged from “1” to “9” before the comparator. Therefore, if the  $k$ -value is “1” or greater then the output image will simply saturate. Different comparator settings were implemented, including having no comparator before adding a bias signal into the  $k$ -value. This was determined through a trial and error process until the output from the DSP Builder model matched the MATLAB test comparison images to help eliminate design errors within the DSP Builder GUI.

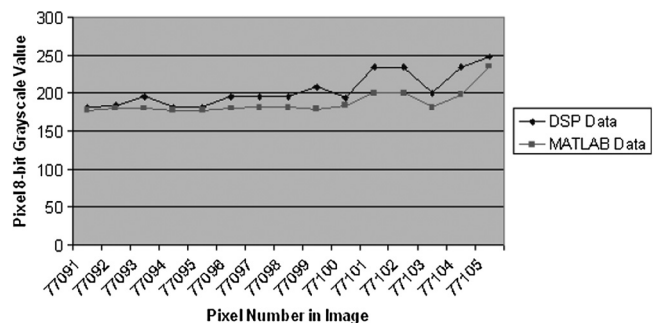
For size comparisons, the binary SMO system is virtually identical (especially the dilation system) to the binary MO system<sup>1</sup> with the only difference being the comparator block used to compare the value obtained from the LHS of Eq. (4) to the  $k$ -value. The erosion circuit is slightly larger due to the circuitry required to calculate the value obtained from the RHS of Eq. (3) for the comparison stage of the system. Again, complexity is very similar to the standard systems, and so the soft system is better since it can do everything the standard system can and more, at the cost of an equivalent or slightly larger size. For the grayscale system, however, in either case if the separated- $k$  or multi- $k$  system is used, the size is approximately eight times larger for the SMO system than the MO system.<sup>1</sup> This is primarily due to the fact that there is a block similar in size to the original system for each  $k$ -value. There is then a decision block required due to there being no other feasible way in

order to reduce the size of this any further. However, all the original functionality of the system is retained and the benefit of noise immunity is the advantage that should outweigh the slight increase in size.

For the advanced operations, the results show that the DSP Builder systems and MATLAB checks for the  $k = 1$  systems are identical, with all of the changes occurring in all the same places as in the previous tests. There are differences and similarities between the images within the grayscale dimension. The differences in the image were far easier to spot and the user could tell a marked change in the pixel intensities for each individual operation, especially when using the “imagesc” command in the MATLAB environment rather than the “imshow” command.

The similarities were that the holes in each of the images were in the same place as in the basic operations. The differences are that some holes were eliminated and appeared to have been smoothed, and small areas like the edge change on the chest of the cat were fused together. These are all classic signs of closing and thus proving the system works as expected. When the  $k$ -value is increased to  $k = 7$  for example, the opening operation output image had completely saturated. Investigation calculations were performed and the right-hand side of the equation for erosion  $\geq k \cdot \text{Card}[B_1] + \text{Card}[B_2] - k + 1$  always gives either “5” for the SE4 case or “9” for the SE8 case, since the core of the SE is only one pixel and this is counteracted by the  $-k$  term in the equation. Also, as the  $k$ -value increases the maximum possible value on the left-hand side of the equation ( $k \cdot \text{Card}[A \cap (B_1)_x] + \text{Card}[A \cap (B_2)_x]$ ) becomes gradually greater until when  $k = 7$ , where it can reach a value of “11” for the SE4 case and “15” for the SE8 case, which is far above the values of “5” and “9,” respectively, thus the image set has a far greater chance of saturating at higher  $k$ -value  $s$ . The SE4 case gave more highly depleted output images because of the fewer number of active pixels inside the SE.

Figure 9 and Table 1 show the details of the comparison process that took place to confirm that the DSP Builder output file matches the MATLAB check image produced by the Image Processing Toolbox inspired m-file. Figure 9 shows the strings of pixel values produced by the “reshape” command in MATLAB. This command takes data from a matrix (i.e., the image) and presents it as a data string that can be plotted. From this graph it can be seen that the two sets of pixel values are virtually identical and follow the same trends and slopes to within a slight variation. This variation is caused by the fact that the MATLAB system has more processing power, can implement loops within the



**Fig. 9** Plot of pixel intensity for the DSP Builder system and the MATLAB check to show the similarity of pixel values.

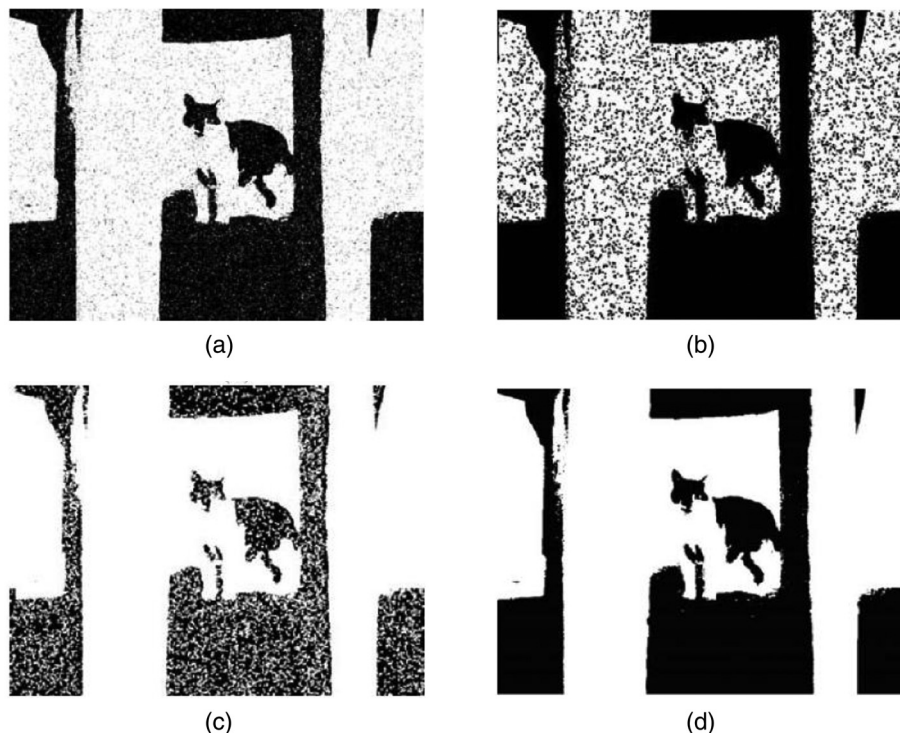
**Table 1** Image comparison tolerances summary.

Match	Binary Basic	Binary Advanced	Greyscale Basic	Greyscale Advanced
100	75	69	77	70
95	–N/A–	–N/A–	79	72
90	–N/A–	–N/A–	89	72
85	–N/A–	–N/A–	93	74
80	–N/A–	–N/A–	96	79
75	–N/A–	–N/A–	95	91

system more easily than the DSP Builder system (hence why they are avoided if possible) and can handle boundary conditions differently by simply not considering the number outside of the image set, in DSP Builder this was assumed to be “0.” They are also very similar as MATLAB has been used to verify all the results in every stage in the development process from standard to soft morphology and this has obviously led to a trend that the DSP Builder systems developed follow the same trends as the MATLAB checks do. In order to explore this variation in a little more detail, another DSP Builder system was designed to compare the images and produce resultant values to indicate how similar or dissimilar the images were. The system inputs the two images simultaneously via the eight-bit-wide input lines and diverts the signals into a series of blocks that look for either an exact match or a percentage match. The blocks themselves, if not looking

for an exact match, input the DSP Builder image data into adder and subtractor blocks and either add or subtract a constant value to the pixel data in order to give the percentage variation. The constant value ranged from  $\pm 13$  pixels (the number of pixels needed to change between the two images to give the follow percentage match) for a 95% match to  $\pm 64$  pixels for a 75% match. These are then fed into comparators where an AND gate is used to combine the signals so that if both are in the desired range then the output logic “1” from the AND is used to enable a counter. This counter is then divided by the total number of pixels in the image and multiplied by 100 in order to give us a percentage match which is output to the MATLAB environment for checking. As shown in Table 1, this produced some very accurate matches in some cases, and most interesting of all, was how the binary accuracies were in fact lower than the grayscale which was the opposite to what was expected. Upon investigation, this was found to be produced by a slight phase shift incurred when the data is passed through the line buffer convolution kernel in order to produce the  $3 \times 3$  image segment analyzed on each clock cycle and depending on how many are used, can cause a shift of between four and seven pixels in both the binary and grayscale imagery. This is another reason for reducing the number of line buffers in the system and so it not only helps to save resources on the FPGA board, but also allows for more accurate imaging.

For size comparisons, the binary SMO systems are very similar or better than the binary MO system,<sup>1</sup> with the opening approximately 1.25 times the size of the original system. This is due to the gates which calculate the cardinality and comparisons for one of the pixel states in the SMO system being similar to one of the comparison stages which made up the MO system. The binary SMO closing system was very good being approximately 0.125 times the size of the binary



**Fig. 10** Simulation results for noisy binary operations using “salt and pepper” noise. (a) Binary counterpart of original image (b) Morphological closing  $k = 1$  result. (c) Morphological opening  $k = 1$  result. (d) Combination of (b) and (c) for noise removal.



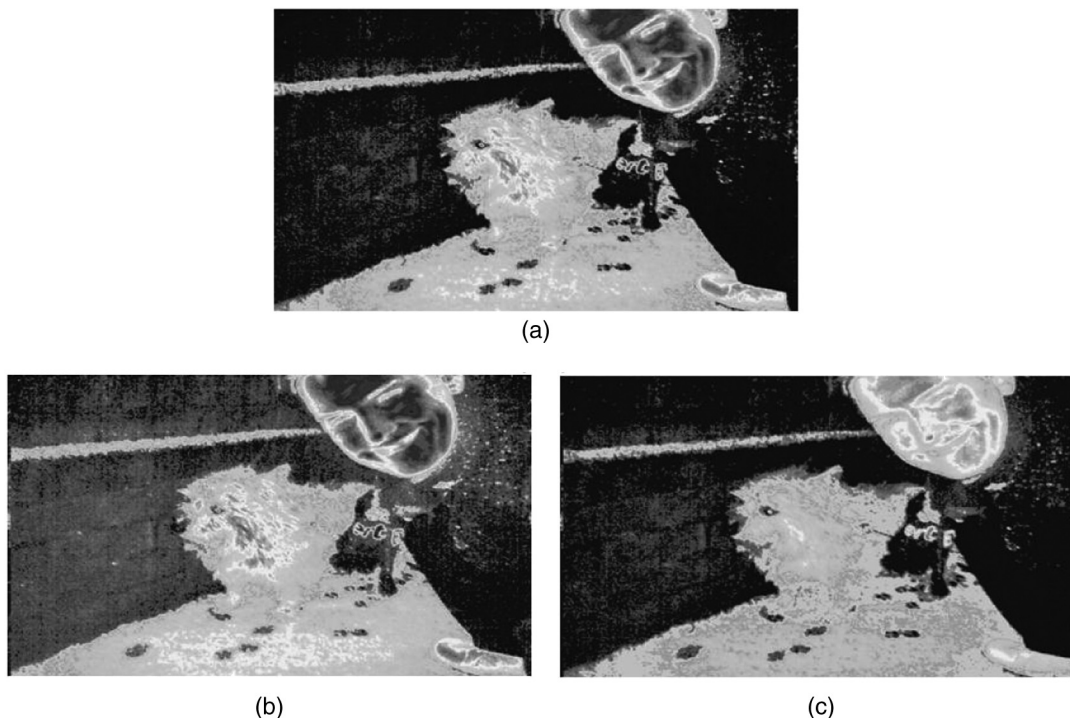
MO system (this being similar to one of the comparison systems) and the MO system used 8 to calculate closing. Since the opening operation is so similar to the basic operation for the grayscale systems, this is again eight times larger. Due to the number of calculations that are performed to develop an “intermediate” result for the closing operation, which is directly fed into a copy of the erosion system, this system is approximately 72 times larger than the MO grayscale closing, which unfortunately is a major drawback.

These sizes were obtained by doing a basic component/gate count and comparison between the MO and SMO systems. Of course, like with the basic operations, all the functionality of the MO systems was retained within the SMO systems.

Various types of noise were added into the image in the form of Gaussian white noise with a constant mean and variance, Poisson noise from the data instead of adding artificial noise to the data, and lastly “salt and pepper” noise. All of the latter was achieved using the built in functions of MATLAB and adding the noise as the data was sent into the DSP Builder simulation via the m-file to simulate as if the noise was captured by a camera. This was accomplished by adding the “imnoise” command in MATLAB into the m-file that sends the data in and out of the DSP Builder simulation environment. When the images were run through the same SMO circuitry, the system produced similar results to the same margin of error as were obtained without the noise. This shows that the noise was eliminated from the white (binary “0”) segments of the image in opening, and for closing, noise was eliminated from the black (binary “1”) segments of the image. This is shown in Fig. 10(a)–10(c). When the two separate images are combined, this completely removes all the noise in the image as in Fig. 10(d).

In order to provide completeness of the coverage of these simulations, Fig. 11 shows the results from the grayscale basic MOs system simulations. Shown in Fig. 11(a) is the grayscale counterpart of the original image whilst Fig. 11(b) and 11(c) show the dilation and erosion operations, respectively. As observed, there are changes between the original image and the two resultant steps, the main changes are that the dilation step causes the wall in Fig. 11(b) to become paler whilst the face and other features remain similar, however, the erosion step in Fig. 11(c) causes the opposite, here the face is paler and the wall remains the same. In all these simulations the  $k$ -value was set to “7” to focus on cases where  $k > 7$  to demonstrate this capability. These checks proved to provide valid results and the resultant imagery is similar to that shown in Fig. 11(b) and 11(c).

In embedded system design, there is a lot of flexibility in deciding which parts of the system should be embedded in the FPGA and which parts should remain as separate ICs. Possibly the most compelling choice is to integrate most of the system into the FPGA chip, including the processor and its peripherals (e.g., bus interfaces, parallel/serial and some of the memory, such as processor cache memory). One of the more vital aspects of the design is in performing Quartus II compilations on the produced VHDL code in order to gain an idea of how much resources these operators would use up when downloaded to a specific Altera device as these operators are designed to become part of a much larger image system which would require a capture device such as a camera which is outside the scope of this material (and hence why a complete computation time for a complete system cannot be included here). The device was changed in Quartus for the compilations by going to the Assignments > Device menu and selecting the appropriate board family. The results of these compilations are summarized in Table 2 for four



**Fig. 11** Simulation results for grayscale operations. (a) Grayscale counterpart of original image. (b) Morphological dilation  $k = 7$  result. (c) Morphological erosion  $k = 7$  result.

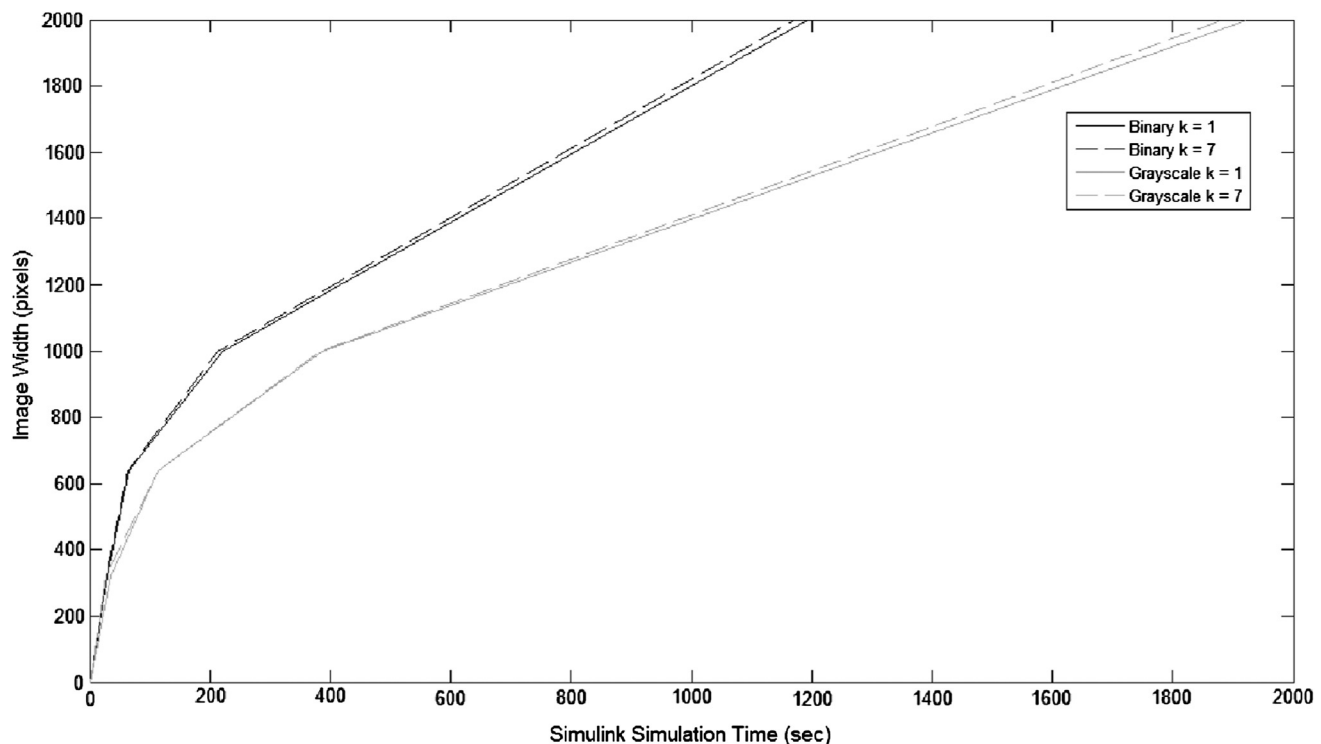


**Table 2** Resource allocations for various Altera board families showing the percentage of logic elements (LE) and pins used.

Operation Description	APEX Family	Cyclone II Family	Stratix I Family	Stratix II Family
Binary Dilate/Erode	7%(2%)	1%(2%)	2%(1%)	1%(2%)
Greyscale Dilate/Erode	Fail (2%)	8%(2%)	8%(1%)	8%(2%)
Binary Open	9%(2%)	1%(2%)	1%(1%)	1%(2%)
Binary Close	>1%(2%)	>1%(2%)	>1%(1%)	>1%(2%)
Greyscale Open	Fail (2%)	24(2%)	48%(1%)	24%(2%)
Greyscale Close	Fail (2%)	Fail (2%)	Fail (1%)	Fail (2%)

common board types: the APEX family using the 20K200EFC series board, the Cyclone II family using the EP2C35 series board, the Stratix I family using the EP1S25 series board, and finally the Stratix II family using the EP2S60 series board. These specifications are without the supporting NIOS processor to assist in the running and real time operation of the system. The APEX board has 8320 logic elements (LE) and 376 pins, the Cyclone II has 33,216 LE and 322 pins, the Stratix I has 25,600 LE and 474 pins, and the Stratix II has 48,352 LE and 355 pins. The latter figures are shown in Table 2, thus the LE usage is the un-bracketed percentage whilst the bracketed number is the pin usage. The pin usage stays the same as the same number of inputs and outputs are used on all the simulations but of course, these remaining pins are vital for when the operators are incorporated into a much larger system. Memory is not included here as each system only used eight memory delays

in order to create the SE convolution kernel and the block type was set to automatic so that it selects the type of memory most practical for implementation from the M4K block memory or M512 block memory used on many of the Cyclone or Stratix series devices so this could differ when the operations are used in a real system. The failures are attributed to the system requiring more LE than there was available on the device and due to the large size for the direct implementation of the grayscale closing operation as no simple direct method could be found. Under these situations it would be a more efficient use of resources to cascade the basic operations together rather than use the direct implementation methodology as the slight increase in memory delays required would allow for an effective implementation of the advanced operations in this case. The binary direct operations are more resource efficient than the standard operations and so this is the better implementation choice. Once

**Fig. 12** Plot to show Simulink computation time against image size.

again, all these trade-offs would have to be based on the end application of the system but either would be appropriate for a dedicated hardware system that could be used as part of another system. This means that the most suitable FPGA to successfully run these types of operations on would be a Stratix II family device or later as this used the least amount of resources.

The last issue to be discussed is the computation time. Figure 12 shows a matlab plot of the Simulink computation time for the simulation stage against the image size. The sizes of the images considered were  $320 \times 240$ ,  $640 \times 480$ ,  $1000 \times 1000$ , and  $2000 \times 2000$ , for an image of size  $4000 \times 4000$ , extra memory would need to be used in the system and so was not included. For the smaller images, the simulation time between the binary and grayscale systems are fairly similar but when the image size increases over  $640 \times 480$ , this is where the two systems begin to diverge and logically, the grayscale system has a larger computation time to handle the eight-bit grayscale data in the images. The interesting point noticed was that the higher  $k$ -value morphological processes had a quicker processing time. It was discovered that this is due to the fact that in the decision making blocks and convergence networks allow the  $k = 7$  data to have a shorter propagation path than for  $k = 1$  thus allowing for a quicker simulation time and system throughput.

## 5 Conclusions and Future Work

This paper has presented and examined a range of SMO structures for both binary and grayscale images for basic SEs. Extensive simulation results have shown that the proposed structures have improved performance over the MATLAB instructions provided by the image processing toolbox that provided the baseline for the comparison and debugging of this work. For higher  $k$ -values, a custom MATLAB m-file was written to help provide the images for comparison purposes as the standard commands that come with the toolbox were not able to do this. The SMO operations developed here are using similar or slightly larger and more complicated circuitry than their MO counterparts. It is easy to interchange image size by altering the image buffer delays and adding more buffers, along with some minor changes to the rewiring and a few constant values, allowing for a more complicated SE to be used if desired, whilst only affecting the gate level of the system. During an in-depth analysis, the system performs the same function to within the same margin of error, given that the image has been either corrupted with noise or not. This was tested several times using various types and combinations of noise and all the results provided the same verification.

The major disadvantages are that no direct implementation pattern could be found for the grayscale closing, resulting in the large increase in size of the system thus making it undesirable to implement on an actual FPGA due to the inefficient use of device resources. Despite this, the advantages are that these systems have increased in size without an associated propagation delay problem that seems to have plagued previous versions of the system. These systems did not suffer as much from this problem primarily thanks to the dummy modules and delays used in their construction.

Future work will consist of developing more application specific systems for various tasks using this graphical

block methodology for robotic-based security systems such as those already developed and undergoing development,<sup>8,10,18,19</sup> but specifically making them work on FPGA systems and with a feasibility study particularly focusing on morphology for the image processing and control of robotic stereo vision systems, as well as continuing the work on implementation strategies and more detailed FPGA resource usages for these developed circuits. Implementation work will continue from the work already completed on basic SISO and SIMO systems comprising all of the respective levels of abstraction to obtain a practical use of the DSP Builder system for the FPGA. This work will then be combined with other work on the use of a system-on-a-programmablechip camera system, controllers of servos and stepper motors, and the development of a wireless communication system, all for implementation on an FPGA so that practical real-world applications can become a reality.

## Acknowledgments

The authors would like to thank the associate editor and anonymous reviewers for their valuable comments and suggestions. Andrew Tickle would also like to thank the Engineering and Physical Science Research Council (EPSRC) for funding this research work during his PhD. Paul Harvey would also like to thank the UK Student Recruitment Office Scholarship (UKSRO) for funding his contributions during his MSc(Eng) and to Altera for their technical support.

## References

1. A. J. Tickle, J. S. Smith, and Q. H. Wu, "Development of morphological operators for field programmable gate arrays," *IOP Electron. J. Phys. Conf. Ser.* **76**(1), 012028 (2007).
2. A. J. Tickle, J. S. Smith, and Q. H. Wu, "Feasibility of a multifunctional morphological system for use on field programmable gate arrays," *IOP Electron. J. Phys. Conf. Ser.* **76**(1), 012055 (2007).
3. L. Koskinen, J. Astola, and Y. Neuvo, "Soft morphological filters," *Proc. SPIE* **1568**, 262–270 (1991).
4. J. Poikonen and A. Paasio, "A ranked order filter implementation for parallel analog processing," *IEEE Trans. Circ. Syst.* **51**(5), 974–987 (2004).
5. C. M. Higgins and V. Pant, "A biomimetic VLSI sensor for visual tracking of small moving targets," *IEEE Trans. Circ. Syst.* **51**(12), 2384–2394 (2004).
6. A. J. Tickle et al., "Image processing virtual circuitry for physics based applications on field programmable gate arrays," *Proc. SPIE* **7100**, 710021 (2008).
7. T. Good and M. Benaissa, "Very small FPGA application-specific instruction processor for AES," *IEEE Trans. Circ. Syst.* **53**(7), 1477–1486 (2006).
8. A. J. Tickle et al., "Feasibility of an encryption and decryption system for messages and images using a field programmable gate array as the portable encryption key platform," *Proc. SPIE* **7100**, 71002N (2008).
9. L. Yang, H. Liu, and C. J. R. Shi, "Code construction and FPGA implementation of a low-error-floor multi-rate low-density Parity-check code decoder," *IEEE Trans. Circ. Syst.* **53**(4), 892–904 (2006).
10. A. J. Tickle, J. S. Smith, and Q. H. Wu, "Feasibility of a portable morphological scene change detection security system for field programmable gate arrays (FPGA)," *Proc. SPIE* **6978**, 69780V (2008).
11. P. Kuosmanen and J. Astola, "Soft morphological filtering," *J. Math. Imag. Vision* **5**(3), 231–262 (1995).
12. P. Dudek and P. J. Hicks, "A general purpose processor-per-pixel analog SIMD vision chip," *IEEE Trans. Circ. Syst.* **52**(1), 13–20 (2005).
13. T. Y. W. Choi, B. E. Shi, and K. A. Boahen, "An ON-OFF orientation selective address event representation image transceiver chip," *IEEE Trans. Circ. Syst.* **51**(2), 342–353 (2004).
14. E. Aho et al., "Block-level parallel processing for scaling evenly divisible images," *IEEE Trans. Circ. Syst.* **52**(12), 2717–2725 (2005).
15. C. C. Pu and F. Y. Shih, "Threshold decomposition of grey-scale soft morphology into binary soft morphology," *CVGIP-Graph. Model. Image Process.* **57**(6), 522–526 (1995).

16. R. B. Fisher, "CVonline: the evolving, distributed, non-proprietary, on-line compendium of computer vision," <http://homepages.inf.ed.ac.uk/rbf/CVonline/> (November 2007).
17. A. J. Tickle, "Applications of morphological operators on field programmable gate arrays," Doctoral Thesis, Univ. of Liverpool, pp. 326–332 (2009).
18. A. J. Tickle, J. S. Smith, and Q. H. Wu, "A grayscale skin and facial detection mechanism for use in conjunction with security system technology via graphical block methodologies for on field programmable gate arrays," *Proc. SPIE* **6978**, 69780Q (2008).
19. A. J. Tickle, P. K. Harvey, and J. S. Smith, "Applications of a morphological scene change detector (MSCD) for visual leak and failure identification in process and chemical engineering," *Proc. SPIE* **7833**, 78330W (2010).



**Andrew J. Tickle** received an ME (Hons) degree in electrical engineering and electronics from the University of Liverpool, UK in 2005 and was ranked fourth in his year. He obtained a PhD degree in the field of computer electronics and robotics from the same university in 2009. His research interests include embedded system development, digital signal processing, security management technologies, autonomous mobile robots, and electronics for astrophysical applications. He worked as an Engineering and Physical Sciences Research Council (EPSRC) PhD, and also as a research fellow from 2009 to 2010 whilst serving as a university teacher at Liverpool from 2009 to 2011. He is currently a lecturer of electrical/electronic engineering at Coventry University and a nonexecutive director of Crosby Communications PLC and Crosby Systems Limited. He is a chartered engineer, chartered physicist, and is currently a member of the IET, IEEE, IOP, and SPIE, serving on several conference and technical committee panels.



**Paul K. Harvey** graduated from the University of Liverpool in 2005 with a BE (Hons) degree in electrical engineering and electronics before completing a certificate in welding and fabrication in 2006. He later returned to the university in 2007 for his MSc (Eng) in telecommunications and micro-electronic systems. His research interests include complex plasma systems, embedded system design and simulation, and computer communication systems. He is currently working as a freelance engineering consultant.



**Jeremy S. Smith** graduated from the University of Liverpool and was awarded his PhD in 1990. Since then he has taken teaching positions as lecturer, senior lecturer, reader, and professor at the same university. From 2006 to 2010, he was the academic dean and later the vice-president of Xi'an Jiaotong Liverpool University. His academic research interests are development of vision-based sensors and control systems to be used in advanced robotic systems for automated welding and other manufacturing processes, development of advanced digital systems based upon system-on-a-programmable chip devices and parallel processing systems, development of neural network and fuzzy logic techniques to allow the fusion of data from a number of sensors to provide control information. Other areas of research include the development of a networking system based upon the CAN bus, the development of robotic manipulator controllers, and the development of an advanced remotely operated underwater vehicle (ROV) requiring the integration of vision and manipulator systems along with sophisticated navigation techniques.



**Q. Henry Wu** obtained an MSc (Eng) degree in electrical engineering from Huazhong University of Science and Technology, China, in 1981. From 1981 to 1984, he was appointed lecturer in electrical engineering in the university. He obtained a PhD degree from the Queen's University of Belfast (QUB) in 1987. He worked as a research fellow and senior research fellow in QUB from 1987 to 1991 and lecturer and senior lecturer in the Department of Mathematical Sciences, Loughborough University, UK from 1991 to 1995. Since 1995, he has held the chair of electrical engineering in the Department of Electrical Engineering and Electronics, the University of Liverpool, U.K., acting as the head of the Intelligence Engineering and Automation Group. Wu is a chartered engineer and a fellow of IET and IEEE. His research interests include adaptive control, mathematical morphology, neural networks, learning systems, evolutionary computation, and power system control and operation.